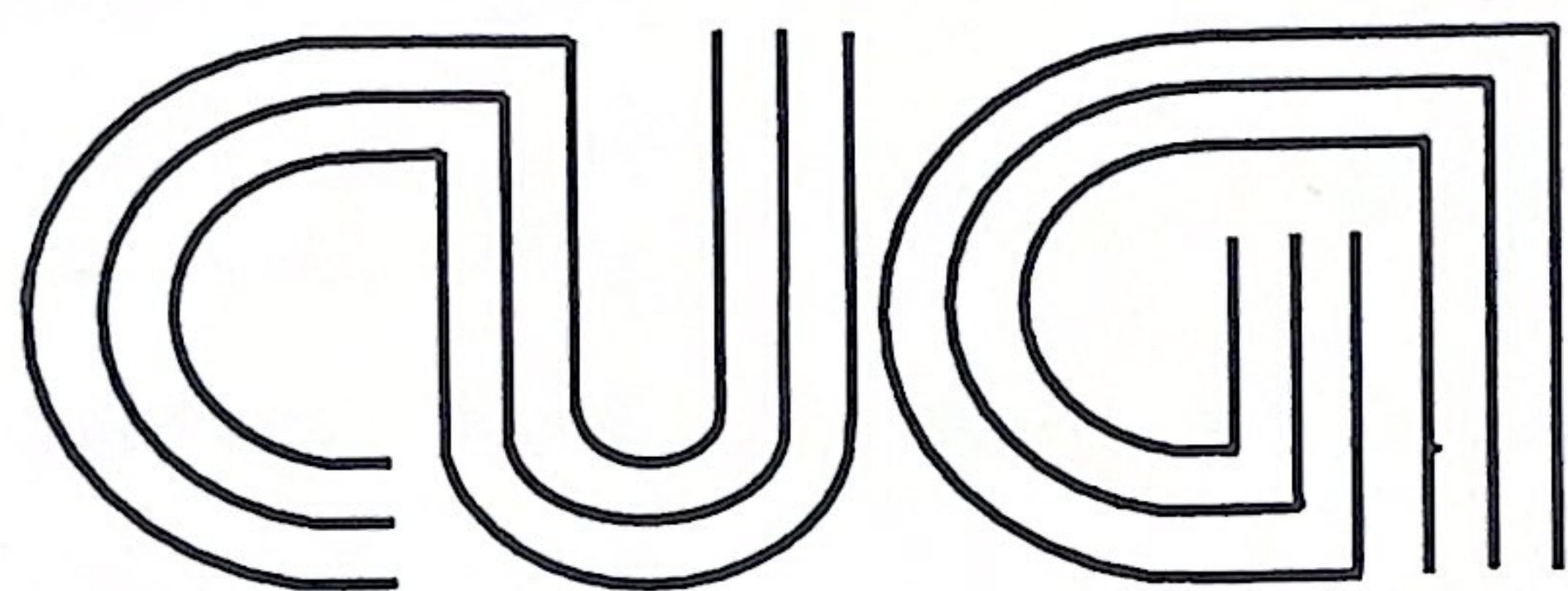




Newsletter

October 1993

In this issue...

*Offline Readers Explained
Picasso II Reviewed
Professional Page 4.0
Optimising Your Code
GVP's I/O Extender*

HONORARY CUGI OFFICIALS

Chairman	Thomas Rolfs
Treasurer	Karl Jeacle kj@broadcom.ie
Secretary	Aidan Hollinshead
Librarian	Magnus Byne jhbyne@unix1.tcd.ie
Technical/Purchases	Matt Brookes
Communications	Colin Dalton csdalton@unix1.tcd.ie
Newsletter Editor	Eddy Carroll ecarroll@maths.tcd.ie

CUGI meetings are held every second Friday, at 8:00 PM in the computer room at St. Andrew's College, Booterstown. All correspondence should be addressed to:

CUGI
c/o St. Andrew's College,
Booterstown,
Blackrock,
Co. Dublin.

If you have a modem, call CUGI BBS at (01) 283 2829. You can reach CUGI by Internet mail at cugi@ieunet.ie, or individual committee members at the network addresses listed above.

All articles in this newsletter are copyrighted by the author unless otherwise stated. If you wish to reprint an article, write to its author c/o CUGI at the above address.



Newsletter Contents

Volume 5 Number 4
October 1993

Editorial	<i>Eddy Carroll</i>	2
Chairman's Chatter	<i>Thomas Rolfs</i>	3
Professional Page V4.0	<i>Ben Lewis</i>	5
GVP I/O Extender	<i>Karl Jeacle</i>	7
Amiga Offline Readers	<i>James Ruane</i>	10
Picasso II	<i>Eddy Carroll</i>	13
To Whom It Concerns	<i>Tommy Gibbons</i>	22
Classified Ads	<i>Secretary</i>	22
Optimising Your Programs	<i>Christian Murphy</i>	24
Thanks for the (Video) Memory	<i>Geoffrey J. Reeves</i>	29
CUGI Library	<i>Magnus Byne</i>	30
Solution To CUGI Crossword #9	<i>Aidan Hollinshead</i>	32

Editorial

Well, a new year and a new committee—it seems that change is in the air. You can read all about the exciting events (such as they were) that led to the current arrangement of members on the committee, but in the meantime, it's nice to see that some things can still be relied on (such as the newsletter coming out a month late).

As you can see from the inside front cover, the new committee can now be contacted electronically—thanks to some work on the part of Karl, CUGI now has its own Internet network address (cugi@ieunet.ie). Most individual committee members can now also be contacted electronically; again, the addresses can be found on the inside front cover.

Elsewhere, I've reviewed VillageTronic's Picasso board—it certainly played a big part in getting this issue out, and it's probably one of the most useful items I've added to my Amiga in the past few years.

Jamie also has a useful introduction to offline readers, which is essential reading for anyone who is a regular caller to BBS's (either locally or abroad). I know that since I started using an OLR recently, my average call time has dropped drastically, and I now rarely use more than one Telecom unit per call, even if I login during peak hours.

Also in this issue, Christian takes a very useful look at the techniques that can be used to speed up your code, while Karl waxes lyrical about GVP's new I/O card. There are even a few words from a certain ex-committee member that some of you may recall.

Meanwhile, I'm still waiting for my shiny new CD³² to arrive. I feel a Tommy Gibbons-style saga in the making here, but we'll have to wait and see. At least it seems to be doing well for Commodore—sales are reportedly good, and occasional Commodore ads have even been spotted in the national press and on TV. Particularly nice is their current ad slogan: "To be this good will take Sega ages." Maybe this will be the time that Commodore finally get it right.

For the moment, let me remind you that the next newsletter is scheduled for January, so early submission of articles would be most appreciated! You can give articles to any committee member, so no excuses about not being able to find me.

To finish off, it just remains to wish everyone a Happy Christmas on behalf of the new committee. See you in the New Year!

E.C.

Chairman's Chatter

by Thomas Rolfs

Well it had to happen some day I guess. I mean, things can't stay the same forever, they have to change eventually. I am of course talking of Geoff's long and industrious reign as CUGI Chairman. We have all been so used to Geoff being up there at the front of meetings, coordinating proceedings with his usual flair and humor, that it now seems somehow wrong that he should no longer be at the helm.

Unfortunately Geoff currently has other more pressing demands on his time which make it impractical for him to tend to CUGI business on top of all his other commitments, and so he has had to stand down this year. Nevertheless Geoff assures me that he will still be attending meetings as usual—it's not like he died or anything. I would like to take this opportunity, on behalf of all past and present CUGI members, to thank Geoff for his extraordinary commitment to the club over the years. What a guy! (Smoke me a kipper, I'll be back for breakfast.) Geoff, I hope you have lots of fun with your new Picasso board.

This paragraph, however, I wish to curse the very name of Geoffrey Reeves Esq. Once upon a time I, like many others, was quite content to just sit back at meetings and simply watch the proceedings. Boy, are those days gone! With Geoff and other members of the, let's call it the *old* committee standing down this year a cry went up for new blood to carry the flame. Feeling guilty for always having sat back and let others do the work, I put my name forward for election. Little did I know what this entailed; but more of *that* story later.

First, let me properly introduce the new committee. This year, there are seven committee members as opposed to last year's eight. Three are from the outgoing committee and the remaining four are new. The veterans are Eddy, Aidan and Magnus. Each has continued on in the roles they held last year: Eddy as newsletter editor, Aidan as secretary and Magnus as Librarian.

The neophytes are Karl, Colin, Matt and myself. Colin is the new Communications Officer whos main responsibility is the running and upkeep of CUGI's BBS. Matt assumes the title of Technical/Purchases Officer and is responsible for sourcing all CUGI hardware purchases as well as dealing with any hardware problems such as repairs. If members are thinking of investing in new hardware they should talk to Matt first.

Election Action

Which brings us to the last two positions on the committee, namely Chairman and Treasurer. When the elected members met to form a committee it was very clear that none of us were eager to take on the role of Chairman. Indeed, after an hour of discussions, we were still no closer to filling the position.

Eventually it came down to Karl and myself for the two posts; all the other posts had been allocated. I had wanted to be Treasurer, but Karl was reluctant to be Chairman by default. The rest of the committee told us that we should decide between ourselves who took which post, as they had no preference either way.

We told them that no, they would have to decide who they wanted as Chairman and that we would stand by their decision; and so a secret ballot was taken. Each member wrote their choice on a piece of paper and placed them into the hat—well, a box actually—where upon they were drawn one at a time and read out aloud.

After four draws, the score was tied: two votes for Karl and two for myself. It was down to the last piece of paper to determine who was Chairman and who was Treasurer. There was no turning back now. Only a few weeks before, I was a contented back-seat member, now as my name was read out I knew that meetings would never be the same again: I was to be Chairman and Karl, Treasurer.

Looking Forward

With the committee formed, what of the future? New blood brings new ideas and new perspectives; however, we need your input as well. Remember that the club exists for its members, not for the committee. It's your club, so make the most of it.

To this end we have drawn up a questionnaire to help us to serve you better. You should have received a copy along with this newsletter; if not, get in touch with the club secretary and ask him for one. Please take the time to fill out all the questions as thoroughly as possible. Next issue, we will summarise the findings.

I'll finish up now by mentioning some of the things we have lined up over the coming months. Firstly, CUGI BBS will be moving over to the club's A1200 as soon as a 120 Mb hard disk has been installed—this should happen within a week or two.

Once the move is complete, the board will be online twenty-four hours a day, seven days a week, except when the A1200 is used for meetings on Friday evenings each fortnight. Colin will keep us updated.

Other things to look out for are the Christmas Quiz, with lot of prizes to be won and a good time to be had by all; a comprehensive comms evening to cover everything you need to know about modems and the like; Amiga networking; CD32 in action; and lots, lots more. It looks like being a pretty packed year, so stay tuned.

Professional Page V4.0

by Ben Lewis

As its name implies, *ProPage 4.0* is aimed at the professional market, and is one of the most powerful desktop publishing programs available on the Amiga. Desktop publishing is basically the ability to lay out a page in a newspaper style, using columns, different font sizes, graphics, etc.—things that you could not normally do on a word processor such as *Protext*.

Admittedly there are word processors which have some desktop publishing abilities, such as *Wordworth 2* and *Final Copy II*. However, they do not usually offer the flexibility and power associated with dedicated desktop publishing packages.

ProPage has been around for some time on the Amiga. The old version was v2.1, which came out a few years back. Version 3.0 came out towards the end of last year, only to be superseded within a few months by version 4.0, which supports the AGA chipset.

This is not the only difference—v4.0 is nearly double the size of v3.0, and offers a host of improvements. The most recent release is v4.1; however, I am still using v4.0, which suffers from a number of bugs here and there—this means that frequent saves are necessary to avoid losing all your work to a system crash caused by the program. That said, it is usually relatively stable.

Features

ProPage sports a number of different frills which make it what it is. It supports PostScript output, which is necessary for having a document printed up by a professional printing bureau. PostScript is a language which is widely used by printing bureaus for outputting to imagesetters, which can give you 2500 × 2500 dpi (dots-per-inch) resolution! This is a bit more than your average laser printer, which can normally only manage 300 × 300 dpi. *ProPage* also supports Pantone colours, which is another standard that is widely used.

Many desktop publishers require you to enter text either directly on screen or by use of an external text editor. ProPage supports both of these methods, but also has its own text editor built in, and supports a direct link to it. It also supports links to other Gold Disk products, such as ProDraw, a structured drawing program, and has its own built in paint program for touching up pictures and clipart.

ProPage can use Adobe Type 1 fonts by converting them to its own format with a supplied program; Compugraphic fonts can also be used. It supports ARexx links, which means that a program can be executed to instruct ProPage to lay out a page in a predefined format. This is useful if you want to setup a particular layout that will be used frequently. ProPage also comes with a large collection of *Genies*, pre-defined ARexx scripts that carry out common DTP tasks for you.

Conclusion

All in all, ProPage is a very powerful, easy to use program that yields excellent results with a little time and effort. It supersedes its main rival, *PageStream* 2.22, in a number of areas. For example, *PageStream* doesn't even support keyboard shortcuts! *PageStream*'s advantage is that it does not take up quite as much processor time and memory space as ProPage, and its screen update is much faster.

A word of warning here, by the way: in order to run ProPage successfully you should have at least 4 Mb of RAM, with 6 Mb recommended. A hard drive is essential, and a fast processor is preferred, such as a 68030 or 68040.

ProPage 4.1 is available from most suppliers, and will set you back about Stg£129 or so. A point to note is that *PageStream* 3 will be coming out soon, and this is heralded as being the best yet—even better in some ways than *QuarkXPress*, the leading DTP program on the Macintosh! Its retail price will be about Stg£250. There is an upgrade offer available to recent *PageStream* 2.22 buyers free of charge, and to ProPage 3 & 4 owners for about £125.

GVP I/O Extender

by Karl Jeacle

With the arrival of a 'multimode data controller' to connect my Short-wave Radio to the Amiga, I found myself continuously leaning around the back of the computer struggling with serial cables to switch between the controller and my modem. This became tiresome pretty quickly, and the need for a second serial port arose.

The only serial expansion board I was aware of for the Amiga range was Commodore's 7-port A2232 board. Unfortunately, this board seemed to have been discontinued and only supported speeds up to 19200bps. I searched through several UK magazines for alternatives but found nothing. Eventually, I came across a discussion of GVP's I/O Extender on Usenet. There were two or three alternatives, but they were quite a bit more expensive, and GVP owners seemed to be happy with their purchases. I looked no further.

It was all going so well...

I ordered my I/O Extender from Amigaman in the USA. They charged US\$109 for the board and US\$25 for shipping and insurance. My credit card was charged a total of IR£98.50. The only annoyance I found here was that I had to fax a copy of both sides of my credit card to them; a little strange I thought, but it seems to be a reasonably common practice.

Three weeks later, the board arrived. Not having purchased a GVP product before, I was a little surprised, but impressed with the quality of the packaging, and general presentation of the board.

Unfortunately, all was not well. The board initially failed to function correctly. The serial ports would open, but I couldn't send or receive data. To cut a long story short, I found that the crystal which was supposed to clock the serial ports had come loose from the board during transit and was stuck at the bottom of the anti-static bag.

I rang both GVP and Amigaman. GVP told me my dealer would replace the board and sure enough, Amigaman sent one out the day I called. A predictable three weeks later, a new board arrived—this time with a slightly less loose crystal but working perfectly.

The Hardware

The GVP I/O Extender is a full length Zorro II card which offers two high-speed DB9 serial ports, a DB9 MIDI port and a standard parallel port. The board itself has the first serial port and the parallel port fitted. A rear

slot cover bracket is supplied with the second serial and MIDI port. Two ribbon cables run from these to the board. You don't have to fit these extra ports if you don't want to, but since I had no other cards in my machine, there was no reason not to. A CPU slot cover plate is also supplied (for use on A1500/2000 machines) which avoids using up a valuable Zorro slot by allowing you to mount the extra ports at the CPU slot.

Another nice touch is the jumpers on the card which allow the serial ports to be run as null modem ports (by simply crossing the Tx & Rx lines). This eliminates the need for a null modem cable on the outside. However, since these jumpers are on the board itself and not accessible from the outside, you'd want to know for sure if this was all you wanted that port for.

Another jumper is provided which switches the parallel port from Amiga Parallel to IBM PC Parallel. The difference here is that the Amiga port supplies 5 volts on pin 14 (allowing devices such as samplers and digitizers to be powered).

The MIDI port on the board is not a standard MIDI DIN connector. You need to buy an additional MIDI expansion unit, which provides two 16 channel MIDI buses. Each bus provides 1 IN, 3 OUT and 1 THRU connectors.

Although the I/O Extender only has two serial ports, you can install as many I/O Extenders in your Amiga as you have free Zorro slots. Each port is assigned a unique ID number allocated 'down' the expansion bus. As you count cards along the bus, you get SerUnit0, SerUnit1, SerUnit2, SerUnit3 and so on. This of course applies to the parallel ports as well.

There is also an empty chip socket and option connector on the board which are reserved for future expansion. I've read on Usenet that there are mumblings of Ethernet capability for this port, but I have my doubts.

The Documentation

The manual supplied with the I/O Extender is one of those small plastic ring-bound affairs. It's 44 pages long and has plenty of diagrams, screenshots and examples on how to install the board, and configure it to work with popular software packages. It also has an index and troubleshooting guide. Quite adequate.

The Software

Software supplied with the board consists of a single disk with drivers and several control programs. An installer program is supplied.

Getting your software to use the board can be achieved in three ways. The most desirable method is tell your application which serial or parallel

port you would like to use. Most terminal programs allow you to do this. So to use your modem with your I/O Extender board, you could enter *gvpser.device* in place of *serial.device*, and select unit 0 or unit 1 to choose which of your new serial ports to use.

The second method involves the use of logical device designations *GVP-Par0:* and *GVPSer0:*. These are managed by a handler in the *l:* directory. Some programs allow you to specify your output device in this manner.

The final method is a last resort. If your application software insists on using the internal Amiga ports, GVP have supplied a program called *SetDevice* which runs at startup and can intercept calls to the serial and parallel ports. An Intuition front-end called *GVPIOControl* is provided which lets you decide whether calls intercepted should be passed through as normal or whether they should be redirected to one of the I/O Extender ports.

Although this last system seems a little nasty, it can work well. If you are using something like a digitizer or sampler on your internal parallel port, the software to control this device most likely deals directly with the hardware. By using this software, you can attach a printer to your I/O Extender and make it look like it's attached to the internal parallel port, so none of your software needs to be changed at all.

A third program called *GVPSerial* is supplied. The function of this program is identical to that of the standard *Serial* program supplied in your *Prefs* drawer. It just controls the GVP serial ports instead of the internal port.

Conclusion

I haven't made any performance tests on the board, but there seems to be a reasonable drop in CPU overhead during high-speed serial transfers. How do I know this? Well, not very scientific, I'll admit, but when *Blanker* kicks into operation when the machine is downloading a file, the splines move smoothly around the screen instead of the usual jerky motion.

I've no real complaints about the board. My only gripe would be that GVP didn't supply DB9→DB25 adaptors for the serial port, but perhaps this is asking a bit much.

Overall, I'm happy with the I/O Extender. It does everything I want it to, has a quality feel to it, and is relatively good value for money. I'd have to recommend it.

Amiga Offline Readers

by James Ruane

As you know, Telecom Eireann recently decided to 'rebalance' our call charge system. This means that local calls, which used to be simply expensive, are now exorbitantly priced, especially during peak hours.

This is definitely bad news for BBS users. A typical call to a BBS might last anything up to an hour, especially when you are reading and replying to large amounts of E-Mail. To stop the phone bill mounting rapidly, many users are switching to using Offline Readers, commonly called OLRs.

These are used in conjunction with a program on the BBS which extracts and packs all of your mail into a *packet* which you then download. After you have replied to all these messages offline, you can then call back and upload your replies. Careful OLR use means that calls can be kept as short as 3 minutes, especially if you're using a fast (9600+ baud) modem.

In addition, using an OLR means that the BBS is free for longer, so that others can call. You can also use your favourite text editor, instead of being forced to use a cryptic BBS editor. Finally, you can spend longer reading messages because you aren't worrying about connect charges.

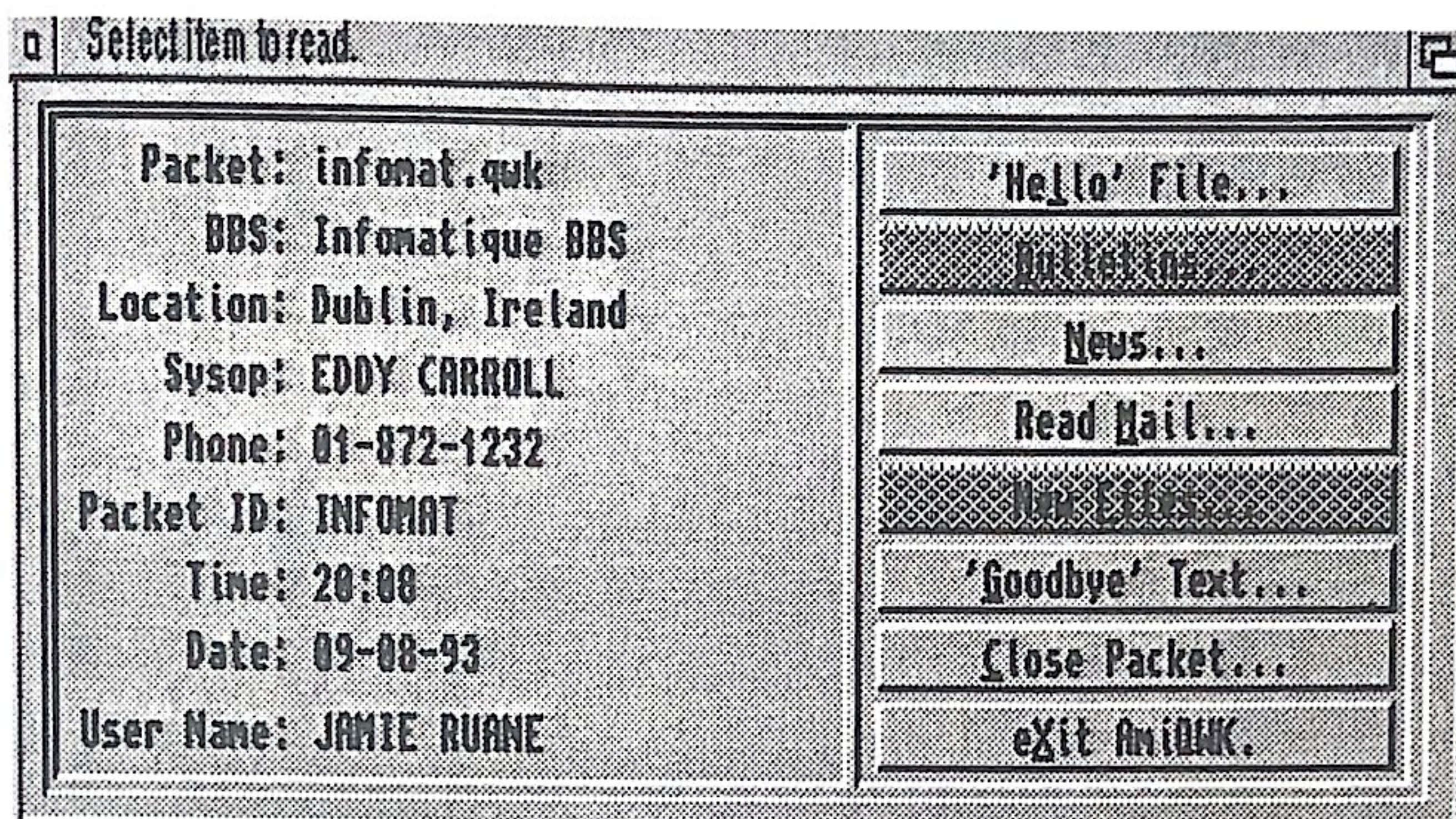
The OLR door

The BBS handles offline readers in the form of a *door*, or external program, which the user enters. This program extracts all the unread messages, and then bundles them into a packet using *LhA* or *Zip*, then allowing the user to download them.

CUGI BBS supports this, with a few added advantages. As it is running on an Amiga, you can use the BBS while your mail packs, playing a game of *Hack & Slash* or downloading a few files. The packed mail is sent to your private file directory, so you could even logoff and download it the next day, if you wanted to.

When you have replied to all these messages (offline, of course) you can then upload a reply packet. The OLR door reads this packet, and then places the replies in the relevant BBS message areas. To other users, there is no difference between these messages and those written online (except for the lack of typos, perhaps!).

There are a couple of standard offline reader formats: XRS (or Xpress Response System), which is mainly used on the PC; *BlueWave*, which I have never seen used anywhere; and QWK, which seems to be the most popular on the Amiga. CUGI BBS supports the QWK standard, as do most other Amiga bulletin boards in Dublin.



QWK packets contain useful information about the BBS itself

The OLR reader

To access the mail packet you just downloaded, you need a reader. On the Amiga, there are three main contenders:

- *Q-Blue*
- *AmiQwk II*
- *Offline Orbit*

All of the above can be downloaded from CUGI BBS. They all support QWK, with *Q-Blue* additionally supporting the BlueWave format, although you're not likely to use this function. There are other readers available to cater for different OLR formats, such as *AmigaXRS*, which handles the XRS format, but as I haven't seen a BBS with XRS support, I have no use for it. Some other QWK readers are *MessageView* and *DaRead*, but I found them to be so buggy as to be unusable.

In my opinion, *AmiQwk II* is the best of the bunch, because of its superior user interface, although it does seem to have a few bugs, including a missing `CloseWindow()` command that occasionally prevents the screen from closing! I have found its bugs to be tolerable, though they would deter me from paying the shareware fee. Both *AmiQwk* and *QBlue* are shareware, with their (American) authors wanting about \$15-\$30. *Offline Orbit* is freeware, so you can get into offline reading fairly cheaply.

Using the OLR is easy enough. When you select *AmiQwk's Open Packet* gadget, it shows you a list of all QWK packets in your download directory. Choosing a packet unpacks it to RAM: using *LhA* or *Zip* (the OLR will

Infenatique BBS -- sysop, EDDY CARROLL

From: EDDY CARROLL
To: JAMIE RUANE
Subj.: RE: ADVERTISING CONSOLES
Area: General

Date: 09-30-93 02:02
Msg#: 4562 <- 4564

#1 of 37

In a message dated 28 Sep 93 19:40:00, Jamie Ruane wrote:

JR> Yes, but it's no use if you already have a SCSI card! A better idea
JR> would be to release a SCSI/Akiko/CD32/DSP card!

Who says if CBM release a CD-ROM for the A4000 that it will be SCSI? The

Akiko
which
stand

JR>
JR>

CD32
contr
defin

Eddy

Writing a reply

From: JAMIE RUANE	Add XX>
To: EDDY CARROLL	Public
Subj.: RE: ADVERTISING CONSOLES	
Area: General	
SAVE	Edit
Cancel	

Close Reply Write Area List < Prev. > Next

Posting a new message in Q-Blue couldn't be easier

automatically detect the archive method used and handle it accordingly).

You can then navigate the message base using the keyboard and mouse. When you reply to a message, your favourite text editor is loaded, with the message you are replying to already loaded for quoting. When you are done, AmiQwk pops up a status window allowing you to modify such things as message header, recipient and so forth. You can even append a *tagline* to your message. (A tagline is a one-liner at the bottom of your messages).

Both Offline Orbit and Q-Blue work along similar lines, but I think AmiQwk is easier to get running and is nicer to use. Also, I found that Offline Orbit deletes all files in the working directory, which I had set to be the RAM disk.

Whilst none of the Amiga OLRs is perfect, they all do their job, and help reduce the amount time you spend online, and thus your phone bill. If you're serious about bulletin boards, you should check them out.

The Picasso II Graphics Card

by Eddy Carroll

The first computer I ever owned was a Vic-20. At the time, it was great: colour, sound, a whole 5K of memory (3.5K usable), and of course, its most memorable feature—a screen display of 22 columns by 23 rows.

Now, ten years on, I have a VillageTronic Picasso II graphics board installed in my A4000, and my Workbench screen has a resolution of 1024 x 768, enough to comfortably accomodate 20 of my old Vic screens. Times have changed.

I first encountered the Picasso at the Orlando Developers Conference last January. VillageTronic had a system set up in the computer room, and were demonstrating Workbench running at an obscenely high-resolution. What impressed me at the time was that it seemed to have no trouble running standard Intuition-based software on its high-resolution screen. No fanfare or flashing lights—it just worked.

Even more impressively, it could transparently run older incompatible software on the same display, by passing through the native Amiga video output. All the other boards I had seen at that time assumed you had two monitors handy, one for the graphics display and one for the Amiga.

Workbench 2.0 Makes It Possible

Much of the seamless integration of the Picasso with the Amiga's operating system is a result of the monitor support Commodore added to Workbench 2.0. Whereas with Workbench 1.3, programs typically supported only a few fixed screen sizes, Workbench 2.0 introduced the display database, which provided a general way of describing any possible screen resolution.

The result of this is that during the three years since 2.0 was first released, more and more software has been upgraded to take advantage of this feature, and as a result, can take full advantage of the Picasso's high resolution modes. Typically, programs will either clone the current Workbench screen mode, or more usefully, offer a screenmode requester that allows you to choose from a list of available resolutions.

Under Kickstart 2.0, Intuition will only allow you to display 16 colours per screen; under 3.0 with Picasso, you can have 256 colour screens, even on the pre-AGA machines. In addition, the Picasso has an onboard blitter to speed up common operations such as text scrolling and area fills.

Once the Picasso is installed, a whole host of new resolutions are available to choose from. With the 1 Mb model, you can display 24-bit colour at 640 x 480, 16-bit colour at 800 x 600, 256 colours at 1152 x 900 and 16

colours at 1600×1200 . With the 2 Mb model, the resolutions stay the same but each resolution can display more colour (from 800×600 in 24-bit up to 1600×1200 in 256 colours).

In the indexed colour modes (up to 256 colours) a standard VGA palette of 262,144 colours is available. The 16-bit mode offers a fixed palette of 65,536 colours and the 24-bit mode gives the full range of 16,777,216 colours.

There are two other variables to consider when talking about resolutions: scan rate and refresh rate. The Picasso supports monitors with scanrates of 15 kHz up to 64 kHz, and refresh rates from 50 Hz to 90 Hz, but not all at the same time. Here's a table showing the refresh rates you can attain for each scan rate at the more common resolutions:

	640 x 480	800 x 600	1024 x 768	1280 x 1024
15 kHz	50 Hz	n/a	n/a	n/a
31 kHz	59 Hz	50 Hz	74 Hz (int)	55 Hz (int)
35 kHz	68 Hz	56 Hz	87 Hz (int)	63 Hz (int)
38 kHz	71 Hz	59 Hz	87 Hz (int)	67 Hz (int)
48 kHz	71 Hz	75 Hz	60 Hz	87 Hz (int)
57 kHz	71 Hz	82 Hz	70 Hz	87 Hz (int)
64 kHz	71 Hz	82 Hz	79 Hz	90 Hz (int)

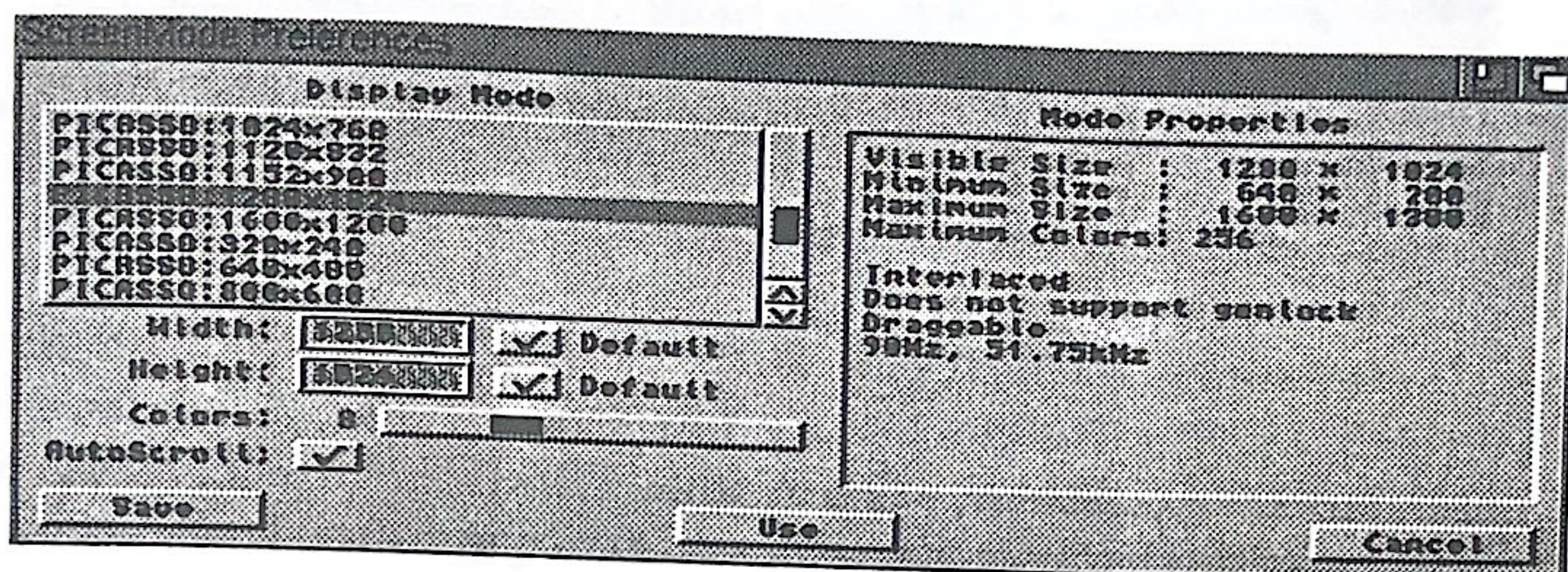
The interlaced modes of 87 Hz and 90 Hz are actually very steady, and are certainly worlds apart from the familiar 50 Hz flicker normally associated with the Amiga. In addition, the 24-bit modes are slightly more restricted in their refresh rates, due to bandwidth limitations. For example, at 800×600 in 24-bit, the refresh rate is 50 Hz.

VillageTronic have said they plan to release a monitor configuration utility that will allow custom modes to be programmed. For most users however, the standard modes will be more than sufficient.

Installation

The Picasso comes well packaged in a box containing the board itself (a full length Zorro II card), a pass-through video cable, a manual, three diskettes, and a stick-on Picasso label, to add to your computer case (a nice touch). The board looks very professional, and features sockets for up to 2 Mb of RAM (so that 1 Mb owners can expand their board later).

A jumper is available on the board to choose between a 2 Mb and 64K memory map; the former is faster, but occupies two megabytes of Zorro II expansion space, which might be a problem for A2000 owners with lots of



Well-written software automatically supports the new screen modes.

memory. The 64K option doesn't use any Zorro II expansion space, but certain features of the board (such as direct access to the hardware from user programs) becomes restricted. On Zorro III machines like the 4000, there are no memory problems, so I was quite happy to leave the jumper at the default 2 Mb setting.

My current Amiga system is an A4000/040 populated with a bridge-board, SCSI card, and A2320 display enhancer to enable me to view the 15 kHz Amiga video modes on my Hitachi SuperSync 19" monitor. I installed the Picasso into my one remaining Zorro slot, connected the output of the display enhancer to the video input on the Picasso using the supplied cable, and finally connected my monitor cable to the Picasso's video output.

And that was the hardware installation... a five minute job.

The next step was to install the Picasso software. Commodore's standard Installer utility was used to control this, which made the process just as painless as the hardware installation. During the install process, you can choose to install optional drivers for Art Department Professional, Image F/X, ImageMaster, Real 3D and Reflections, along with some picture display programs and other utilities. These allow access to the 16-bit and 24-bit display modes not supported by Intuition (which only allows up to 256 colours).

Once the appropriate files had been copied to my hard disk, it only remained to choose the scan rate of my monitor (64 kHz) from the list provided, and I was all set. One quick reboot later, and the Picasso was up and running.

At first, something seemed wrong: garbage was appearing on the screen, and some of the scrolling didn't appear to work. A quick visit to the manual cleared things up however; I was running my utility *CpuBlit* which uses the CPU to perform certain blitter operations. Since *CpuBlit* isn't intended

to work with foreign graphics boards, it was causing trouble. The manual recommended that I remove it, and I did, at which stage all the problems disappeared. (Time to work on an update, I think.)

First Steps

After rebooting, the next port of call was the Workbench ScreenMode preferences utility, where the list of available screen modes now included eight new Picasso resolutions, ranging from 320×200 to 1600 × 1200. I settled on 1024 × 768 at 79 Hz, in 8 colours. One quick click later, and my screen area had literally doubled. Not only that, but the 79 Hz display was rock-steady, a breath of fresh air after using DBLPAL at 50 Hz for the past twelve months.

Time to return to the manual, for a more in depth look at the features available. This turned out to be very well written, with all the supplied software and options clearly explained in good English (VillageTronic is a German company).

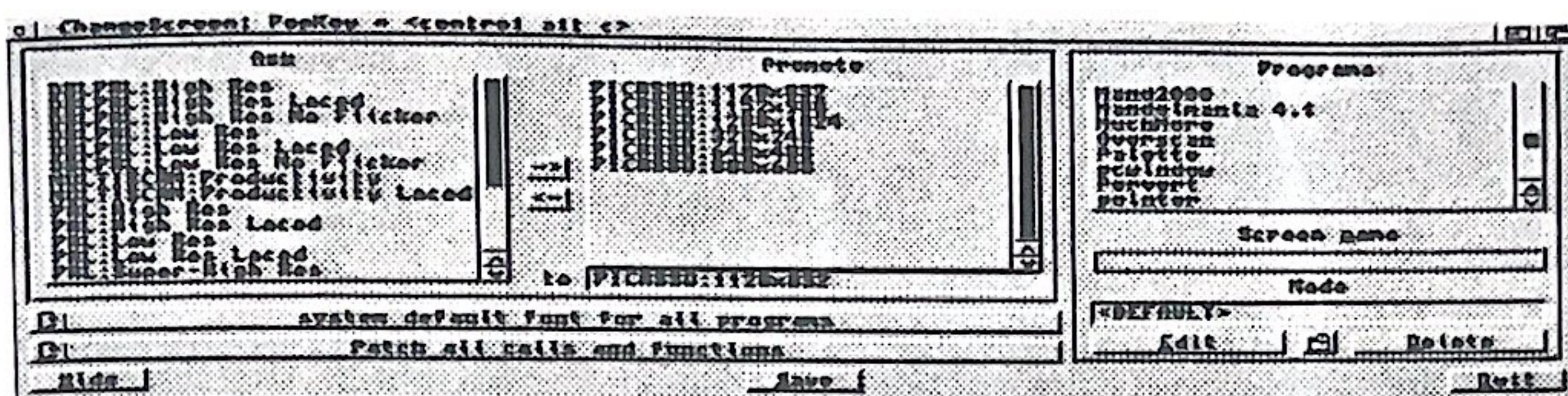
I discovered that the Picasso requires three main files in order to operate: *village.library* which resides in the Expansion drawer, *vilintuisup.library* which goes in Libs, and *Picasso* which is stored in the Monitors drawer. With these three files present, Workbench automatically detects and configures the Picasso as the startup-sequence is executing.

Another important part of the Picasso software is the *ChangeScreen* commodity. This utility intercepts any program that tries to open a new screen and gives you a chance to promote the screen mode from that requested by the application to one of the new Picasso modes (or even a different Amiga resolution). This is particularly useful if you have programs that don't yet fully support the display database. *AmigaGuide* help is provided for *ChangeScreen* which is a nice touch.

You can choose to make the promotion temporary or permanent. The latter causes *ChangeScreen* to store the details in its database so that when you run that program in future, you won't be bothered with a requester. You can also tell it to never promote a particular program, essential for some applications like *DPaint* that use programming techniques incompatible with the Picasso.

A final option is *Copy Continuously* which causes programs to think they are still running on an Amiga screen, while in the background, the Picasso software continually copies the screen into Picasso video memory. This is intended for programs that are particularly incompatible, but in practice, I haven't come across any that it worked well with.

At any time, pressing Ctrl-Alt-C will call up the *ChangeScreen* configu-



ChangeScreen allows old software to work with Picasso screen modes.

ration window, where you can access the program database to add, delete or edit entries. You can also use this window to force all requests for a particular screen mode (say Super72:800 × 600) to be promoted to a Picasso mode without ever giving the requester.

Picasso in Action

So much for setting everything up; time to try out a few applications. The first thing I fired up was the AmigaTeX previewer, with its screen promoted to 1024 × 768. This is a nice viewing size because it allows an entire page to be viewed in one go yet still remain readable. No problems at all—it worked fine.

Trying out some other programs yielded similar results. *Lucy*, the offline reader I use for Cix, ran quite happily at 800 × 600 (83 Hz). Even the excellent mandelbrot generator *Mand2000*¹ ran flawlessly, complete with real-time zoom effect.

Final Copy II presented a problem. At first sight, it appeared to work fine in high-resolution modes, but when I tried to adjust a margin on the ruler bar, the system locked up. The same proved true for a Solitaire game, when I tried to move a card with the mouse. It seems that any application that makes use of BOBs (Blitter Objects) will suffer from this problem, until VillageTronic add support for it. (The exception is Workbench, which allows disk icons to be dragged with no problems at all.)

However, this is not quite as bad as it seems—all that happens is that these programs can't currently take advantage of the new Picasso screen modes. They work fine when Picasso is told to leave them alone.

I ran into a different problem with *Imagine*. Although the object editor appeared to run okay on the Picasso screen, the mouse pointer seemed unable to reach the buttons on the left edge of the screen. I eventually figured out that there is a problem when using mouse pointers with a hotpoint

¹ Bruce Dawson, the author, currently has a Picasso and is hoping to include direct support for its true-colour modes in the next release.

which is not at the left edge of the pointer image—the image can never move past the left edge of the screen, so if the hotspot is in the middle of a cross-hair, there will be a dead area of up to 16 pixels that becomes inaccessible.

I find it difficult to believe that this is a limitation of the video hardware, so it will hopefully be corrected in a future software update. VillageTronic regularly release updates to fix reported problems—I've received two in the last two months—so this is unlikely to be a problem for long.

I also chose not to promote a few other programs, mainly those which used a screen display of 640×256 (such as my terminal program). Because all of Picasso's new screen modes have a 1:1 aspect ratio (approximately), such programs can come out looking squashed. Future versions of the software may add support for 2:1 aspect ratio modes.

The final quirk I noticed was that the mouse pointer seems to be slightly magnetic. This sounds strange, but what I mean is that as I move it up and down the screen, there is a slight bending of the image. This is definitely a Picasso artefact and not a monitor problem, since it happens even when the cursor is invisible. In any case, it's unobtrusive and not particularly noticeable unless I'm feeling bored.

Screens Galore

One of the most impressive features of the Picasso is its ability to support multiple, *draggable* Amiga screens. While other boards can allow several screens to be opened simultaneously, each screen typically occupies the entire display. On the Picasso, you can drag down the frontmost screen to reveal an application running behind it.

Of course, since the Picasso is only based on a standard VGA chipset, it can't do all the clever tricks the Amiga's Copper does, but it makes a good effort. The colour palette will be set to match that of the frontmost screen, as will the number of bitplanes of any background screens. This means that if you drag down an 8 colour screen to reveal a 16 colour screen behind it, the colours may appear distorted.

In addition, you can't drag screens with more than 16 colours (presumably because changing from chunky mode to planar mode on the fly is a bit beyond the capabilities of the hardware).

There is another limitation as well: only one screen may appear in the background at a time. However, an unexpected bonus is the ability to view Amiga background screens. Unlike the Picasso screens, which allow you

to move the mouse into them and interact with them² Amiga screens are read-only; in fact, the Picasso takes a snapshot of the screen at the point when you last switched away from it, and displays that instead. It's not quite as good as the real thing, but useful nonetheless.

It's difficult to describe, but there's something very satisfying about being able to drag my high-resolution Workbench screen up and down in the same way as I always have—it makes it very easy to forget that I'm running on a third-party graphics card and not on a prototype of AAA.

True-colour Support

Of course, if all the Picasso allowed you to do was run Workbench at a higher resolution than before, it mightn't appeal to everyone. Happily, there is also copious support for true-colour. The feather in Piasso's cap is undoubtedly *TV Paint*, which is available at £190 for the full version, or £49 for the junior version. (Even including the cost of the Picasso card itself, this still works out cheaper than buying a stand-alone copy of *TV Paint*!)

Enough has been written about *TV Paint* elsewhere; suffice to say that it is far and away the best 24-bit painting package available on the Amiga, and it is widely used throughout the video industry.

If, however, you don't feel like spending more money, fret not. Bundled with the board are three picture viewers: *ViewIFF*, *ViewGIF* and *ViewJPEG*. These three will allow you to view pictures in any of the three supported formats, taking full advantage of the 24-bit colour modes, even under Workbench 2.0. I tried this out on some JPEG and IFF24 files I had handy, and the quality was stunning.

VillageTronic also include a front-end to the three viewer programs, which can be taught to recognise file formats automatically. As a CLI person, I didn't find much use for this but it seemed like a nice idea.

I also had an opportunity to try out the Art Department Professional and Image F/X drivers. In the case of ADPro, the Picasso appears as an option on the list of savers, and selecting Save produces a requester allowing an appropriate true-colour mode to be chosen. Once chosen, the image appears on the screen. I found 1152 × 900 in 16-bit to be particularly impressive.

With Image F/X, the same resolutions are available, but they are accessed through a custom Render module instead. This makes it easy to view the current image you're working on at any time, by simply clicking a

² In fact, this ability to interact with background Picasso screens is only present under Kickstart 2.0 and 3.0. Under Kickstart 3.1, the Picasso currently doesn't support interacting with background screens, due to some internal changes in the Amiga's libraries.

button. With both packages, the cursor keys can be used to scroll around the image if it's too big to fit onto your chosen screen.

As source material. I used some PhotoCD images I had handy; the quality was excellent. While such images can look very nice in HAM8, seeing them in true 24-bit was a revelation.

Roll Your Own

Also included in the VillageTronics software package is a support library which allow you to write your own programs to display 24-bit images on the Picasso. This really does make it as painless as possible to access the true-colour capabilities. Some example C code is provided to demonstrate various features of the board.

Other utilities included in the package are *Styx*, a simple 24-bit screen blanker, and *PicassoSwitch*, a commodity that lets you switch the hardware between the Amiga and Picasso displays at the touch of a hotkey (useful for software that bypasses Intuition to generate its own display).

An Intuition benchmark utility is also included to give you some idea of how much faster the Picasso is than the native chipset, but the results of this were a little difficult to interpret (lots of numbers, but no indication as to how they related to each other!). The final part of the Picasso pack is a basic paint package called *Personal Paint*. Its main feature is that it supports the Picasso screen modes in up to 256 colours.

The Need for Speed

At this point, you're probably wondering what the catch is. Well, the catch (such as it is) is that Picasso requires a fairly powerful machine to get the most out of it. I certainly wouldn't recommend anything less than a 68030-based system, because updating high-resolution screens can be very CPU intensive, even with the Picasso's blitter support.

The other essential component is memory—the more the better. On my own system, I have 18 Mb, which is certainly more than adequate. A 10 Mb system would do nicely, but if you only have 6 Mb, don't expect to be able to open too many screens at the same time.

The Picasso allocates all its screens from fast memory where possible. While this is good, in that keeps your CHIP ram free, it does mean that your main memory tends to get eaten up quite quickly, especially once you start dealing with 256 colour or 24-bit screen modes. If all you you want to do is run Workbench at a high-resolution however, 6 Mb should be plenty.

Speedwise, the Picasso suffers in comparison with some of its competitors by only supporting Zorro II, and not Zorro III. This doesn't mean it won't

work in Zorro III machines—plainly it does—but it does limit the maximum transfer rate of data into video RAM to about 3 Mb/second.

This is probably most noticeable when flicking from one 256 colour high-res display to another, as the entire screen contents have to be copied to the video memory. In practise though, it usually takes my monitor longer to resync to the new screen than it does for the Picasso to perform the copy, so it's fairly unobtrusive.

The real speed test comes in day to day use. As a former 8 colour AGA Workbench user, the Picasso feels significantly faster, particularly when scrolling text. If I move up to 256 colours on Workbench, there is a definite slowdown, but performance is still much better than 256 colours under AGA (even in the 15 kHz AGA modes). Other cards may turn out better benchmarks, but at the end of the day, the Picasso should be fast enough for most people.

Competition

And speaking of other cards, what of the Picasso's competition? Currently, there are several other boards priced under £700, and offering similar features to the Picasso. These include the Piccolo, Retina, Merlin, and EGS.

The EGS and Piccolo both use the same video chipset as the Picasso, however they also support Zorro III for faster data transfer. Rather than write their own graphics routines (as VillageTronic have done), these two cards use the EGS retargetable graphics system from Vionna Development.

EGS offers a complete replacement for Intuition, including features such as full 24-bit support, automatic dithering on screens with fewer colours, real-time window dragging, and lots of speed. On the negative side, some people find it a bit ugly (myself included) and of course, software has to be specially written to support it. While EGS does offer Workbench emulation as well, my experience with this on the Piccolo in September was that it has some way to go before it was as transparent and stable as the Picasso.

Conclusion

I'm convinced I made the right choice opting for the Picasso. While I've mentioned several problems I had with it, it's more significant to note the problems I *didn't* have with it. It was as steady as a rock, and a few days after I installed it, I had already become so used to it that I found Workbench at 640 × 512 looked decidedly clunky.

If you want a board that integrates flawlessly with the system, doesn't get in the way, and provides you with access to much higher resolutions, more colours, and better speed than AGA, then this is the board for you.

To Whom It Concerns

by Tommy Gibbons

Dear Sir,

Whilst listening to the radio the other night, the news that our national airline had attempted to buy another national competitor's airline was related to the general public. This news seemed a little odd at the time as the company was undergoing restructuring due to serious financial difficulties. It was also revealed that the employees of the national carrier had sought or were seeking advice from some people willing to financially aid this company.

It must be stated that the airline executives had already sought and received approval for State financial aid and the licence to relieve a large portion of its workforce of the need to pay any more income tax. The State, in its wisdom, decided to employ the unfortunate former workers as full time unemployed, as well as paying the airline the right to make them unemployed.

If an employer (the State) had employed employees (the airline executives) to carry on certain functions (make money) and if said employees were unable to fulfil that role, it may be more prudent to relieve the employees of the task (being too great for them) and employ someone who was able to achieve a successful employment than to allow said failed employees the task of undoing the damage they had already done to the company.

Is it possible that they may fail? They have done so already. With regard to buying out the competition, it was assumed by one observer that part of the "buying more time" plan was to open a cheaper air route to our overseas neighbours. And in order to do this, it was felt necessary, by the "how do we spend more money" committee, to buy a whole airline company, preferably the strongest national competitors.

It may not have occurred to the "we'll rescue this company from our undoing" committee to drop their own air fares. They have less people to pay, less PRSI contributions, less employee insurance and no company tax as the company is not making a profit. Therefore with a lower employee bill and subsidies from different charitable organisations, it could possible to charge customers less.

Maybe not. Eventually the workforce that was too large in the first place could end up below adequate manning (or womanning) levels and there would be no local competition to show up any irregularities as no comparisons could be made on a local level.

The situation appears similar to that of another company that toys with the idea of selling computers. This company would like to delve into the commercial market but is unable to maintain a mediocre share of the games market worldwide. The technology in the machine is of quality standard, even above some professional standards. Then why is it failing? A joke...

A man walked into the doctors.

"Doctor I am suffering from intense pains all over my body."

"Hmmm, could you show me?" said the doctor.

The man pressed against his rib cage with his index finger and groaned in agony. The doctor was suitably impressed.

"Anywhere else?" he asked

"Well, here," said the man, pressing against his calf muscle and almost collapsing on the floor with pain.

"And here," said he, breathing heavily pressing against the side of his head and immediately releasing an uncontrolled moan of anguish.

"Obviously you have some ailment afflicting your body; I'll do some tests. Come back tomorrow for the results," said the doctor.

The following day, the man returned to the doctor.

"I have found the problem," said the doctor.

"Great!" said the man, "What is the cause of my dilemma?"

"Your right index finger is broken!" said the doctor.

Laugh if you wish but maybe someone at this computer company should go and see a doctor to try and figure out if the patient is worth keeping alive under present conditions, or if the patient can make it by undergoing some organ donations. In the company's present format and method of operation, the grim reaper beckons.

Classified Ads

Wanted: a 17"-20" monitor to display colour resolution up to 1280 × 1024.
Apply to smug person at back of meetings.

Optimising Your Programs

by Christian Murphy

Back in the 8 bit days, you used to see games advertised as being written in "100% machine code". This impressed me, even when I didn't know what it meant, except that it was likely to be 'blazing fast'.

In fact, the choices were limited: you could program in BASIC, assembly language (only masochists write programs in machine code, or even with a hex monitor) or Forth. There were a few other languages, even compilers, but 48K or 64K of RAM was a severe limitation. There just wasn't enough space available for a compiler and a reasonably sized program to be in memory at the same time. Assemblers were just small enough to be practical, and the BASIC interpreter was stored in ROM anyway. The processors were so slow that it was often worth counting cycles for given instruction sequences to find *the* most efficient way of doing something. I well remember tweaking an interrupt routine so that an 8 bit microcomputer could read 9600 baud. In that case, and many others, there *was* no alternative to the fastest way: it wouldn't work any other way.

We are now in the 32 bit era, and the 64 bit days are dawning. Processors are hundreds, or even thousands of times faster than then. The amount of storage space is increased by a factor of twenty to thirty.

Many people treat Amigas as faster Commodore 64s. They want to "get the most out of the machine", and kill the operating system because they can program the hardware better than anyone else. These days they're throwing out half a megabyte of stable, well-tested software. It's sickening the number of programs that duplicate the operating system functions, and do a worse job.

The time between new machines is comparable to the time it takes to make a large program. The variety of Amigas belies the old perception of 500s with 512K RAM, OS 1.3, and original chipset. Users have '020, '030 and '040 processors, ECS, AGA, Retina and Picasso displays with different resolutions and refresh rates, dual floppies, hard disks (SCSI and IDE) and varying numbers of megabytes of RAM. The only practical way to handle this type of diversity is to let the OS handle the details.

The one thing that hasn't improved in the computer industry in the last thirty years is the people. Programmers don't think faster. They haven't got any cheaper. The amount of work they have to do has increased enormously. People require more than just speed. The graphical user interface they want simply needs more code than a punched card interface.

Anything with less than 200 lines of code is trivial, and anything with less than a thousand lines is tiny. And that code has to do much more sophisticated things, like handling mouse input. The best programs impress more for their user interface than their speed. One of the best user interfaces I've ever seen is that of *Mand2000*, a Mandelbrot generator. These programs have been written for years, yet none has had such an elegant interface. That and the 256 colours are what makes me say "Wow", not the speed. Steve Jobs recommended that 90% of a computer's processing time should be spent on user interface. I think that's an exaggeration, but I do believe that maybe as much as 90% of a programmer's time should be spent on it. Certainly it is noticeable how much Gadtools and Gadtoolsbox have improved programs on the Amiga.

The point is that speed is less important than it was. I don't care if your game updates its sprites at 50 Hz if I can't run it on my Amiga. If the game is not interesting, I don't care how super-optimised the code is. Demos with zillions of real-time ray-traced phasar polygons are boring, especially when they look like a guru meditation or a frozen white screen.

Unless you're writing a device driver, use a high level language. If you are writing a device driver, write as much of it as possible in a high level language. There is plenty of evidence (from research), that the number of mistakes that a programmer makes is directly proportional to the number of source lines in the program. In nearly all cases, high level languages are more terse. A well written high-level program is much, much easier to understand than its assembly equivalent, and you don't have to put in as many comments. Poorly documented code is a horror to read, no matter what language is used.

Instead of wasting time writing optimised assembly, concentrate on your design. This means that you should decide what data structures you're going to use and how you're going to manipulate them, *before* you start writing code. These choices are going to affect the speed of your program more than hand-optimising in most cases.

The golden rule is KISS: Keep It Simple, Stupid. In other words, prefer simple data structures to complex ones. Don't keep a list sorted if it doesn't speed things up. Don't use a linked list instead of a table unless there are obvious advantages. Don't use *malloc()* if you know in advance how big your data is going to be. This approach will leave more time later for optimising.

Don't sacrifice generality lightly. The Amiga Exec is a masterpiece of a kernel, one of the best of its time. One of its great strengths is its flexibility,

which is achieved in simple ways. For example, all lists are derived from `struct MinList`. The list manipulation functions can be highly optimised and yet general. The library interface is standard and very flexible.

For the first version of any program, a good clean design is essential. The way to divide the program up is by data structure. Each data structure has associated functions through which it is created, initialised, modified and destroyed. A clean design has clean interfaces for these data structures. The purpose of each function is clear and there are no hidden dependencies between data structures. If your design is good, much of the code will seem almost to write itself. Use small, general-purpose structures rather than large, highly specialised ones.

Fred Brooks said in *The Mythical Man-Month*, "Plan to throw one away, you'll do it anyway." The reason why you'll throw it away will be because it is very difficult to predict how any program is going to behave without running it. When you run it, you'll see all kinds of things that could be improved. You may even see that it is fast enough without any special tweaking. That leaves you time to improve it in other ways.

Even more important is that other people get to see the first version. They'll complain about things you never even thought about. It's very possible that it won't run at all on their machine.

With a bit of luck, the whole design won't have to be discarded. A few modifications will probably be necessary. Therefore the code for the first version should not be too elaborate.

If speed is a problem, the best way to fix it is to go back to the design. Did you choose the best data structures? Would a hash table work faster than a sorted list? If you sorted the googaws, would that make it fast enough?

One helpful cue is the profiler. Many compilers, such as DICE, allow you to compile a program so that it will measure the time it spends in each function. If 75% of the program's time is being spent in one function, it makes little sense to optimise elsewhere. Most programs do spend most of their time in a small section of the code¹ but without a profiler, it is difficult to guess where this is.

The best optimisations are those that reduce the complexity of your algorithm.² It may be worth some analysis to see what the complexity of

¹ That's why caches and virtual memory work.

² The complexity of an algorithm is a measure of how the time it takes relates to the size of the problem. For example, Quicksort is a better algorithm than Bubblesort, because,

the algorithm is. If the complexity is exponential or factorial, that algorithm is not useful except perhaps for small amounts of data (say, less than fifteen items). It may be that a simple change in the data structures may make one of the better known algorithms feasible. A look at Knuth or Foley & van Dam could be well pay off.

Perhaps the basic trade-off in computer science is between space and time. Often it is possible to sacrifice one for the other. You can speed up repetitive calculations by caching results, or even pre-calculate the most common cases. If you have a list that needs to be traversed in more than one order, you could keep lists of pointers to the items in the list. Be aware of special cases: too many of them make you design incomprehensible and your code unreadable. The danger is that you'll end up with a program as useful as a guru-producing demo, and just as hard to debug.

Having considered these solutions (and applied them, if relevant), you can try looking at the code itself. If you are using C, it may help to use pointers instead of array indices. A table of pointers can be a useful form of 'pre-calculation'. The gain will probably be slight, but may be worth it if you are optimising inside a loop that is being iterated many times.

It is rarely useful to re-code a library routine, though it is often tempting. 90% of professional programmers can't program a binary search properly. That's why *bsearch()* is provided. Concentrate your efforts on your own code. However, it is useful to try to reduce calls to library routines, or to batch them. For example, it may be worthwhile to buffer data that is going to be output, and use one library (or system) call for the whole lot. I/O often takes significantly more time than calculation. *WritePixelArray8()* and the buffered stdio routines are library routines written with this type of usage in mind.

The string library functions, *strcat()*, *strlen()*, and *strcpy()*, are an example of a case where hand optimisation can be useful. If your profiler shows a lot of calls them, you can probably save time by, for example, storing pointers to the end of your strings.

If you've done all this, and speed is still a problem, the next preferred solution is to throw some money at it. Buy a faster machine (or an accelerator board) and require your users to have one. Alternatively, consider getting a better compiler (I assume you've already turned on the maximum optimisation level for your compiler). If you're being paid for it (or if there

if n is the number of items to be sorted, Quicksort will on average sort them in a time proportional to $\log_2 n$, whereas Bubblesort will use time proportional to n^2 . Quicksort is said to be an $O(\log_2 n)$ algorithm and Bubblesort an $O(n^2)$ algorithm.

is a deadline), this is often cheaper than time spent hand optimising small pieces of code.

If you still have to hand optimise, general techniques are replacing function calls with in-line macros, unrolling loops and using look-up tables. (Note that many good C compilers, like the Gnu C compiler, have options to generate inline function calls and unrolling of loops.) If after all this effort, the program is still too slow, it's time to consider assembly. The profiler can be used to find the bottleneck that is slowing down the program. Then part of the program can be re-coded in assembly.

Note that the gains from each successive step from changing the algorithm to recoding in assembly are less and less. The improvement of assembly over a good compiler is going to be marginal, and very dependent on your skills as an assembler programmer. Your assembly routine may even be slower than the compiler's output. The program has become less portable and harder to modify or improve. It is not hard to hack a program to shreds by sitting staring at the screen and compulsively coding and re-coding.

The best solution to many programming problems is to take a break. Leave the room, take a walk, do some shopping. Sleep on it. Eureka is not something that only happens to geniuses. The solution can pop into your head while you're waiting for a train, or just before you drop off to sleep. Going back later with a new tactic is very satisfying.

References

Donald E. Knuth, *The Art of Computer Programming*. Addison-Wesley, 3 Volumes:

Vol. 1: *Fundamental Algorithms*, 2nd Ed.. 1973

Vol. 2: *Seminumerical Algorithms*, 2nd Ed.. 1980

Vol. 3: *Sorting and Searching*. 1973

No list of references on programming is complete without these. They're not light reading, but they are comprehensive. Knuth invented the idea of analysing the complexity of algorithms. He also wrote T_EX.

Frederick P. Brooks, *The Mythical Man-Month*. Addison-Wesley, 1975

This is the well-written result of Brooks' experience as head of the team that wrote OS/360, the operating system for the IBM 360 machines of the late sixties. The project was a disaster: it was huge, very late and full of bugs. Although his project involved literally thousands of programmers, the lessons he draws from it (and his previous experience) are relevant to anyone doing serious programming today.

Robert L. Kruse, Bruce P. Leung, Clovis L. Tondo, *Data Structures and Program Design in C*. Prentice-Hall, 1993

The great advantage of this book is that it has lots of good examples written in C. It's well laid-out, with plenty of diagrams. A very good book for people who know a bit of C, but would like to beef up their programming skills.

J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics: Principles and Practice*, 2nd Ed. Addison-Wesley, 1990

The bible for graphics programming. This has descriptions of almost every useful graphics algorithm, plus plenty of background.

Thanks for the (Video) Memory

by Geoffrey J. Reeves

To all those who contributed to the send-off I received on 15 October, a very sincere 24-bit thank you. It was a very pleasant surprise to find that all those I've encouraged, cajoled and slagged off over the years appreciated my efforts. I have always maintained that the club's success is due to all those who did a demo, wrote an article, organised this 'n' that and so on. My good fortune was to be a member of some very good and dedicated committees. Their contribution to the club's success should not be forgotten—I recalled a number of them at the AGM and perhaps you might consider going through your back issues of our Newsletter to remind yourself of those characters.

To the new committee, I pass on my very best wishes—I thoroughly enjoyed the times I spent doing my part. I hope that the 'new boys' enjoy their appointment—I'm very impressed so far and, I reckon, you ain't seen nothing yet!

As I explained at the AGM, other interests have taken over much of my spare time but I fully intend breaking the tradition whereby retired Chairmen disappear. I'll be the bearded old codger sitting at the back of the meetings muttering things like 'Nah...not at all like I used to do it' and 'I remember when 8 bits were a big byte'. The smug grin will be your reminder of a very much appreciated graphics board and personalised paperweight.

Many, many thanks.

October 1993

CUGI Library

by Magnus Byne

The CUGI library has a large selection of Amiga books, including a range of Abacus books covering most aspects of the Amiga, and Commodore's Rom Kernel Manuals (essential if you are doing programming). In addition, there are reference books for the C64 and C128, along with general programming and computer books.

You can borrow books from the CUGI library at any meeting, free of charge. You must return them by the following meeting, but if nobody else wishes to borrow them, you can renew them for another two weeks.

Here's a list of what's currently available (new titles may have been added by the time you read this):

Abacus

AmigaDos Inside and Out

Amiga for Beginners

Amiga Disk Drives Inside and Out

Amiga Guide for Advanced Programmers

Amiga Internals

Amiga Tricks and Tips

Desktop Video Guide

Commodore Reference Manuals

Amiga ROM Kernel, Libraries 2.0

Amiga ROM Kernel, Devices 2.0

Amiga ROM Kernel, Includes and Autodocs 2.0

Amiga ROM Kernel, Style Guide

Amiga ROM Kernel, Libraries and devices 1.3

Amiga ROM Kernel, Includes and AutoDocs 1.3

Amiga ROM Kernel, Hardware Reference Manual 1.3

Using the System Software 2.0

AmigaDos Manual 3rd Edition

The C Language

C Quick Reference

New C Primer Plus

The C Programming Language (K&R), 2nd edition

The C Library

Programming in C

General

Amiga Applications

Beginners Guide to the Amiga

Amiga Format Book

The Amiga System

AmigaDos

The Anatomy of the A590

ICPUG Newsletters

Graphics

Amiga Artist

Amiga Vision

Inside Amiga Graphics

Using Deluxe Paint (V2)

Programming

Mapping the Amiga

Amiga Programmers Handbook

68000 Pocket Book

MC68881/2 Motorola Data Book

Microcomputer Puzzles

Golden Oldies

C64 Programmers Reference Guide

C128 Programmers Reference Guide

C64 Service Manual

C128 Service Manual

Advanced C64 Machine Code

The Anatomy of the C64

The Anatomy of the 1541

Hardware

The CUGI library also has the following hardware available for loan:

- Audio Sampler
- Video Backup System
- Modem (2400 baud)
- Video Digitizer
- Handheld Scanner

You need to be a member of the club for at least six months before you are entitled to borrow any hardware items. For the Audio Sampler and Video Backup System, there is a rental charge of £2 and a £3 deposit. For the other items, there is a rental charge of £3 and a £7 deposit.

For more information contact me at any club meeting.

Solution To CUGI Crossword #9

C	A	R	P	E	T	S			A		A
A		O		M			A	D	O	P	T
U	N	I	F	O	R	M	S			P	
T								P	E	A	R
I	T	C	H	I	N	G				A	
O		O		V				A		I	
N	U	R	S	E			O		P	O	S
E		P					S		P		E
D		O					S	C	O	L	D
	C	R	E	A	M			I			
E		A				A	M	E	N	D	I
B	A	T	T	E	D			T		C	
B		E					V	E	S	S	E

Solution to Aidan's crossword in the July 1993 Newsletter.

Camera ready pages for this issue
were produced using:

Amiga 4000/040
AmigaTEX V3.1h
Sharp Laser Printer

Photocopied by
whoever was available